Productionizing predictive models

Code breakfast

Julian de Ruiter & Ivo Everts

GO DATA DRIVEN

About me

- Background in computer science and computational biology (TU Delft)
- PhD doing breast cancer research at the Netherlands Cancer Institute (NKI)
- Machine learning engineer at GDD

Julian de Ruiter

GO DATA DRIVEN

About you

GO DATA DRIVEN

This breakfast

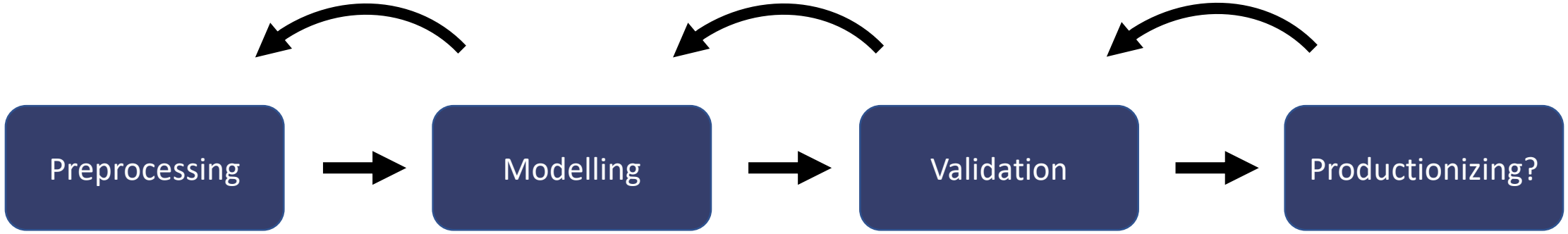
- ~15 minutes introduction
- ~1-1,5 hours hackathon / demo
- ~15 minutes wrap-up
- Plan to finish around 10:00

GO DATA DRIVEN

The machine learning process

GO DATA DRIVEN

The machine learning process



Preprocessing	Modelling	Validation	Productionizing?
Exploring and cleaning the data	Picking the right model	Validating if the model solves the problem	Getting the model where it can be used
~60% effort	~10% effort	~30% effort	100-???% effort

GO DATA DRIVEN

Productionizing ML models

GO DATA DRIVEN

What is productionizing?

GO DATA DRIVEN

Productionizing

- Goal – convert model into a (standard) format that can be run in production

- How – depends on the production environment
  - Re-write code into a (production-quality) library
  - Wrap in an API for interfacing with other components

GO DATA DRIVEN

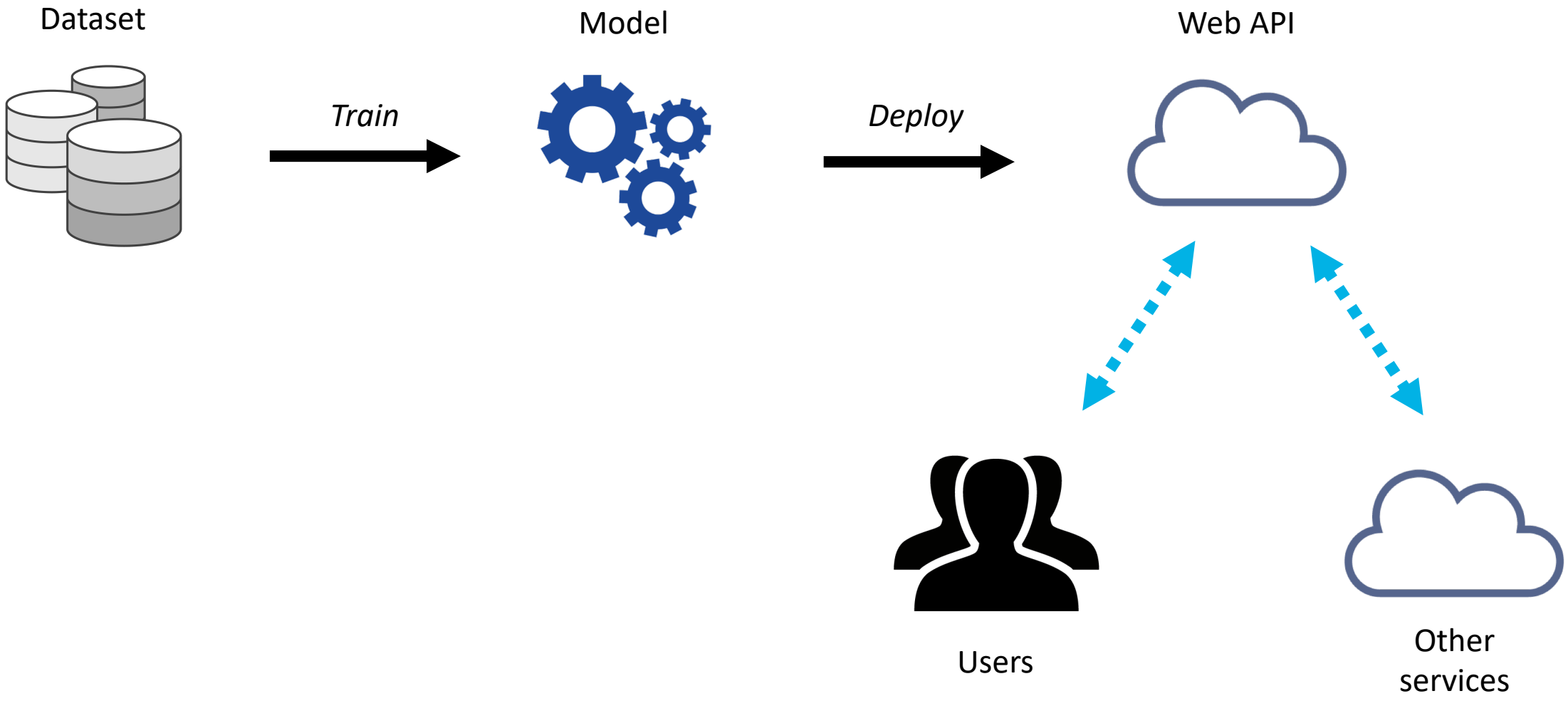
Productionizing

- Follow best practices
  - Version control
  - Code quality checks, unit testing
  - Logging / monitoring

- Consider deployment patterns
  - How will the model learn and predict?
  - What will we expose to the outside world?

GO DATA DRIVEN

Example: web-based API



However, many models look like this

jupyter titanic-model (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Markdown

Loading the data

We can load the dataset into a dataframe by calling `pd.read_csv`. Note that we split our dataset into a train and test set, so that we can do some exploration on our training set and later guage the accuracy of our model on the test set.

In [92]:

```
%matplotlib inline
import pandas as pd

# Read train/eval datasets.
data = pd.read_csv('../data/train.csv')

# Split data into train/test set.
data_train, data_test = train_test_split(data, test_size=0.2, random_state=42)

data_train.sample(3, random_state=42)
```

Out[92]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
373	374	0	1	Ringhini, Mr. Sante	male	22.0	0	0	PC 17760	135.6333	NaN	C
848	849	0	2	Harper, Rev. John	male	28.0	0	1	248727	33.0000	NaN	S
593	594	0	3	Bourke, Miss. Mary	female	NaN	0	2	364848	7.7500	NaN	Q

Visualizing some variables

Visualizing data is crucial for recognizing underlying patterns to exploit in the model.

In [136]:

```
sns.barplot(x="Embarked", y="Survived", data=data_train);
```

GO DATA DRIVEN

How do we move this into production?

GO DATA DRIVEN

How do we move this into production?

- Start building a Python package
  - Isolate main components, move these into modules
  - Identify building blocks -> make reusable functions/classes

- Improve code quality
  - Implement quality checks (pylint) and tests (pytest)
  - Document code (docstrings) and package (readme, etc.)

- Wrap model in an API (Python, Flask)

GO DATA DRIVEN

Hackathon

- Background
  - Client interested in upselling cruise ship tickets
  - Noticed that in the titanic disaster, people in higher ticket classes had a higher chance of survival
  - Would like to present this information during the booking process to sell more 1st class tickets

GO DATA DRIVEN

Hackathon

- Scenario
  - Data scientist has created a model predicted survival probabilities based on the titanic dataset
  - We have been asked to move his/her notebook into production
- Goal – build a documented + tested Python package that exposes the model as a web API

GO DATA DRIVEN

Hackathon

- Getting started
  - Clone our Github repo at https://bit.ly/2ZpNqS4, read the README
  - Setup a clean Python environment and
    install the packages in notebook/requirements.txt
  - Try running the notebook and see if you understand its contents
- Afterwards - continue with the Step 2 (see readme)

GO DATA DRIVEN